EECS 373 - Homework #2

Due 5 February 2024

Name: Yuzhen Chen **unique name:** yuzhench

Due at 11:59pm on 5 February via Gradescope. We'll release solutions immediately after the deadline for use in midterm review. Please use the answer boxes provided. Submit a PDF of your completed assignment to Gradescope. Typed answers and neat handwritten answers are both acceptable.

Question 1

Short answer questions: **[15 points, 3 each]**

A) When the APB Bridge wants to read data from a peripheral, which signal does it use and what is the state of that signal?

PWRITE = 0 for read

B) What is the memory address of the input data register for GPIO Port C (GPIOC_IDR)?

0x48000810

C) What does the GPIO Mode Register (MODER) control?

The mode register (MODER) controls the I/O (input/output) mode of each pin of the GPIO ports

D) Why would an APB bus peripheral need to use wait states?

Since sometimes the peripheral access and data calculation may need time to be ready, so before it is finished, the processor needs to wait for it.

E) Does there need to be a physical memory or register module for each valid memory address on an ARM Cortex-M processor? Justify your answer.

Answer: no need
Reason: there are some addresses are already preserved for memory mapped IO and the value of that addresses can be generated by peripheral devices like FPGA board MODER

Question 2

A) In reference to the GNU toolchain for ARM process, explain in a straightforward manner what the linker does and what it is used for. **[10 points]**

The linker will find the global declarations and references functions or variables in other foles in the object files, and then find a new memory organization for the Data, text, symbolic table to generate a large executable file. Also, the linker is used to link all the object files together, so that when the user changes one file, there is no need to recompile the whole source files.

B) Consider the linker script at http://www.bravegnu.org/gnu-eprog/lds.html. Briefly explain why the label “eoa” is at a higher address than the label “stop” even though the label “eoa” appears first. (Though largely irrelevant to the question, be aware this appears to be ARM but not Thumb assembly). **[10 points]**

“eoa” is in the data section, and the “stop” is in the text section. the linker will set the data section below the text section.

Question 3

Rewrite the following C code function, “A,” in ARM assembly (UAL). You should assume “print” is some ABI compliant function which takes a single integer argument and B is also an ABI

compliant function. (Note, one point of the ABI is to be able to mix C and assembly like this, the linker will make it all work!) **[15 points]**

```
void main(void) {
  int a = 4, b = 3;

  b = A(a,b);
  print(b);
}

int A(int x, int y) {
  int a;
  a = B(x + y);
  print(a);
  return(a + x - y);
}
```

```
    .global A

    .p2align 2

    syntax unified

    .type A, %function

A:

    .fnstart

    push {r4}

    push {r5}

    push {r6}

    move r4 r0 // let r4 = x

    move r5 r1 // let r5 = y

    bl B // branch to the function B
```

```
mov r6 r0 // use the r6 to store the return value from function B

bl print // branch to the print() function

add r0, r0, r4 // a += x

sub r0, r0, r5 // a+x-y, let the return of the function A to be a + x - y

pop {r6}

pop {r5}

pop {r4}

bx lr //return from A to main function

.fnend
```

Question 4: Anemometer Design Problem

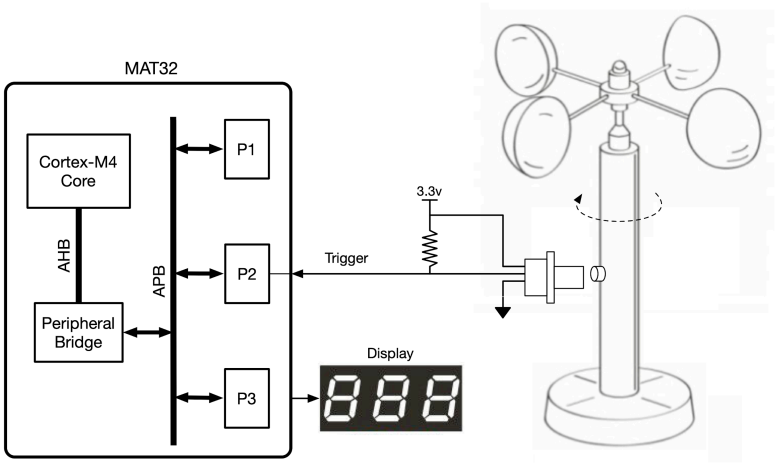
Your task is to design the measurement system for a hand-held anemometer. The device measures wind speed with a rotating cup system. The system measures the rotation period, scales appropriately, and displays the result. Your task is to design the peripheral modules that are connected to the APB

and function properly with an ARM core. You will then write a C function that interfaces with each module that calculates the wind speed.

You will likely find it helpful to read the rest of the problem before solving any of the subparts.

Interface to the APB bus

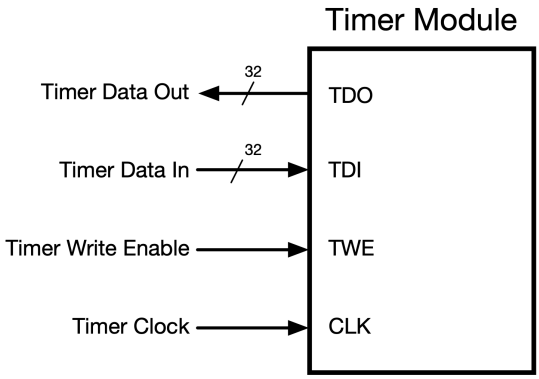
The MAT32F20 is a new ARM Cortex-M4 microcontroller that is identical in nearly every way to the STM32L4 microcontroller. Your job is to design two custom APB peripheral modules for his MCU, an input for the Hall effect sensor and a custom 32 bit timer module. In order to measure the rotational speed of the rotating cups a magnet has been placed on the rotating shaft of the anemometer. When the magnet passes in front of the hall effect sensor a trigger signal is sent to the MAT32.



This fictional MCU has the following APB3 bus interface. The APB bus signals follow APB timing and protocol. Read and write cycles are provided on the next page. PSEL is configured to be “1” when memory locations **0x40050000-0x40050007** are accessed.

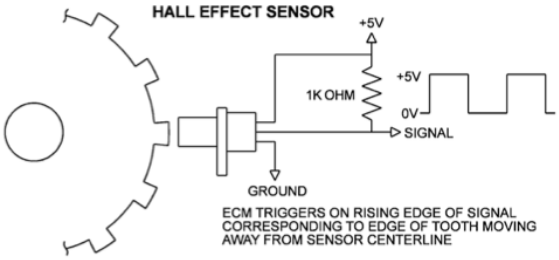
Integrated Hardware Timer

The hardware timer increments the value stored in its internal register on every rising edge of the *Timer Clock (CLK)*. When the *Timer Write Enable (TWE)* is set high, the value on *Time Data In (TDI)* is latched in the timer module on the positive edge of the *Timer Clock (CLK)*. If the counter reaches its maximum value it wraps around to zero (*i.e.,* it is a modulo counter). The current value of the timer is always available on *Timer Data Out (TDO)*.



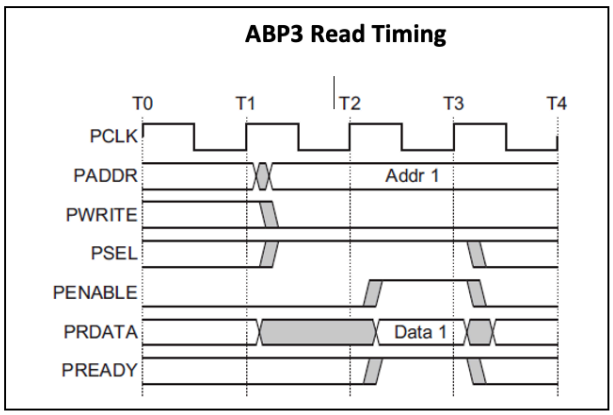
Hall-Effect Sensor - AH1815

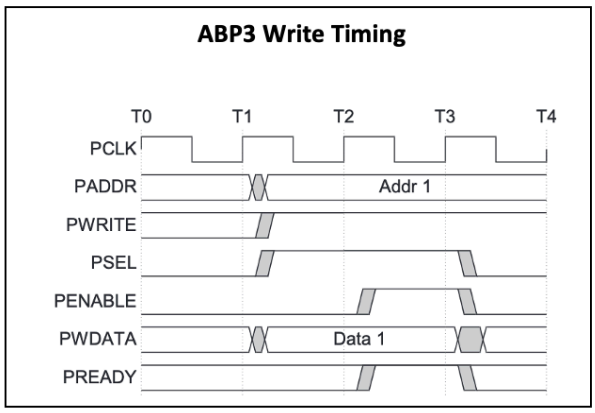
The AH1815 is an integrated Hall-Effect, non-latched sensor. When sufficient magnetic field strength is detected the sensor will output a high signal. Otherwise, the sensor will output a low signal. Moving a magnet near the sensor will cause the output pin to toggle. If a magnet is attached to a rotating shaft a hall effect sensor may be used to measure rotation period.
https://en.wikipedia.org/wiki/Hall-effect_sensor



APB Timing diagram

The following diagrams are provided as a reminder of the APB timing with no wait states

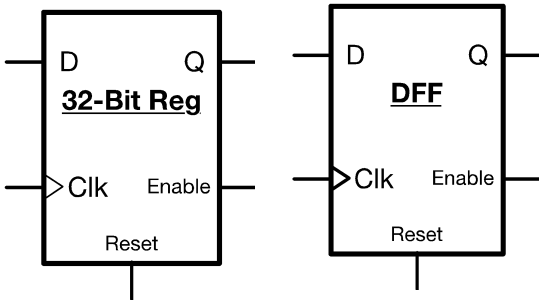




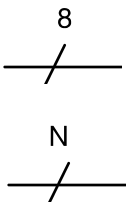
APB Bus Conventions

You may use standard gates such as ANDs, ORs, NOTs (as well as standard bubbles), and DFFs. Be sure to show all connections. You may use GND and VCC to indicate a logical 0 and 1 respectively. You may **not** use Boolean or Verilog expressions. Here are some example symbols.

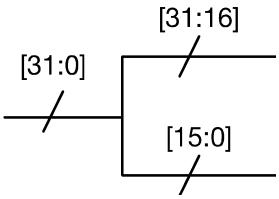
Example DFF and Register (enable is optional)

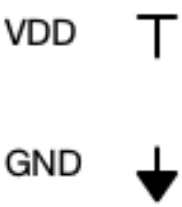


Label the number of lines in a bus.



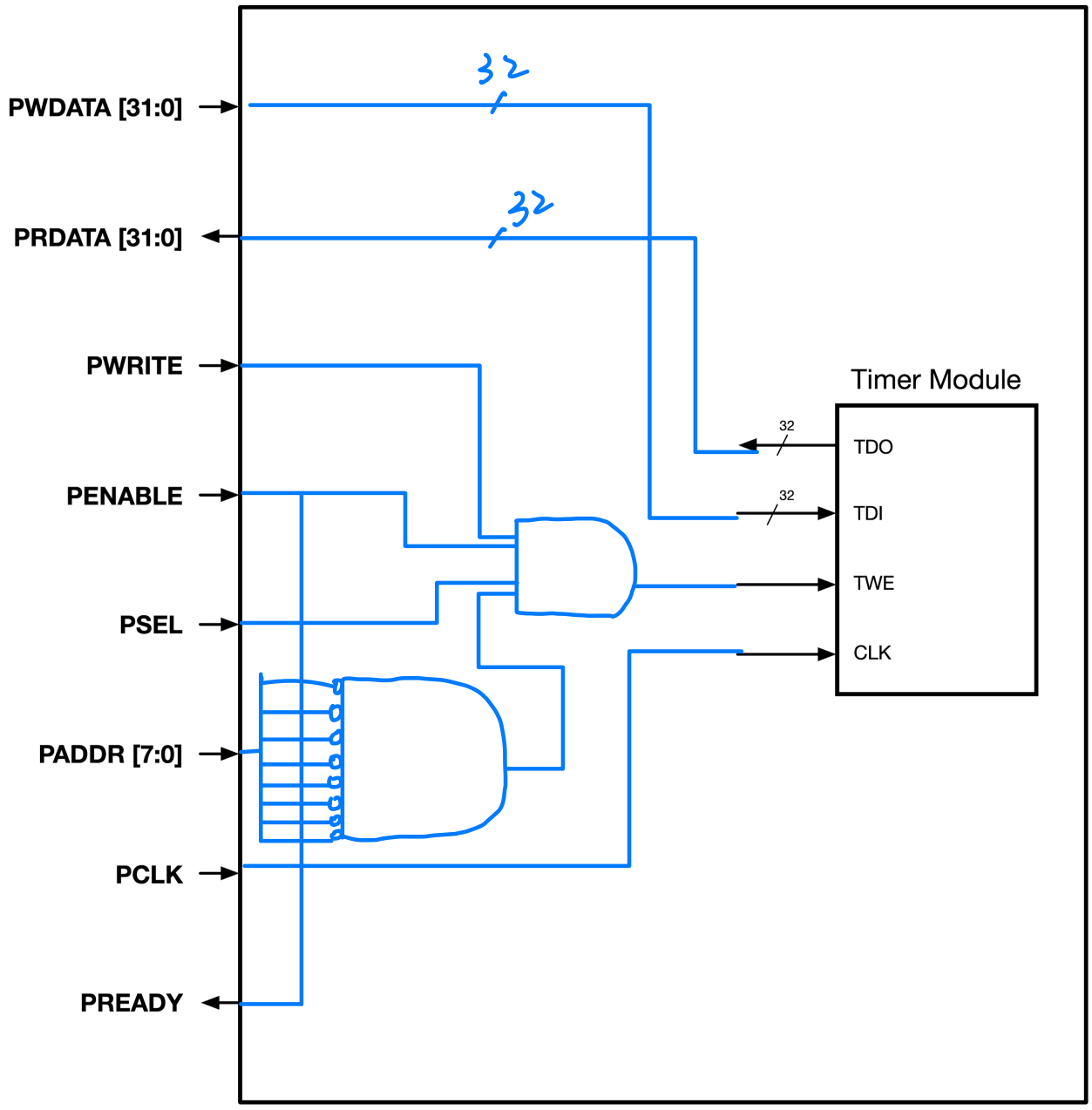
Bus concatenation and data labeling





Part 1: Hardware Timer APB3 Bus Interface (16 points)

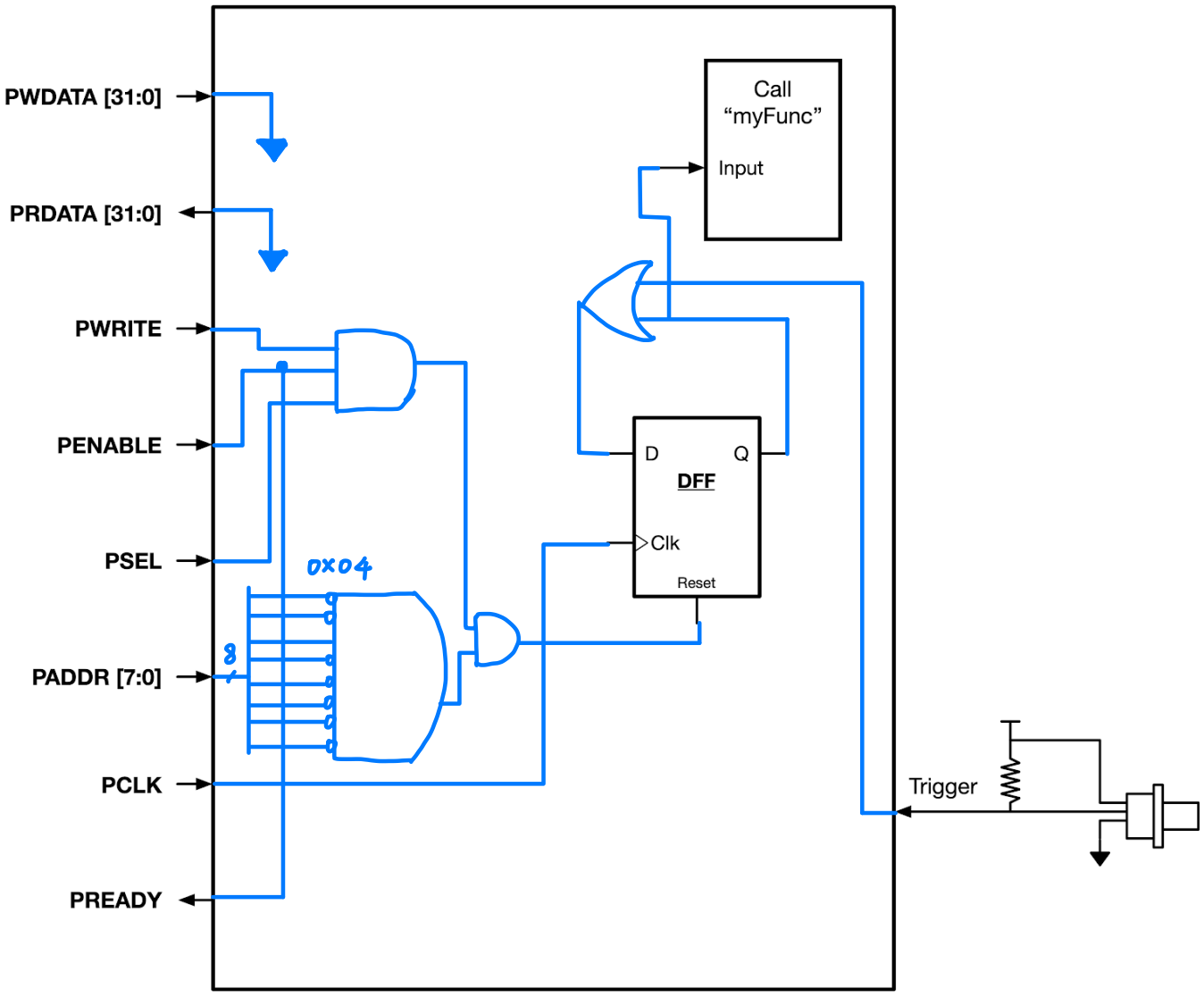
Provide a hardware interface between the APB bus and the hardware timer to allow the MCU to read and write the counter. Assume the timer's counter register is at address **0x40050000** *(and only that address)*. You may use standard gates such as ANDs, ORs, NOTs, and inversion bubbles. Be sure to show all connections. You may use GND and VCC symbols to indicate a logical 0 and 1 respectively. You may not use Boolean or Verilog expressions. You will lose points for superfluous or highly inefficient logic. ***Note:*** We have put all the APB connections on the left. Be sure to drive all outputs.



Part 2: Hall Effect Sensor Peripheral Module (16 points)

Each time a magnet passes in front of the hall effect sensor it provides a high pulse (*Trigger*) that is several clock cycles (*PCLK*) wide and is not synchronized to the APB clock. Matt has designed a special hardware block called “Call myFunc” which will cause the ARM Core to call a function when the input line is held high.

- Design logic using standard gates and D flip-flops that will hold the *input* to the *Call myfunc block* high when the *Trigger* from the hall effect sensor goes high. Your module must hold *input* high until it is cleared
- One D flip-flop is provided, though you may use additional ones as needed.
- Also, provide logic that will clear the high *Input* signal to the *Call myfunc block* when a write to memory address **0x40050004** *(and only that address)* occurs.



Part 3: “myFunc” (18 points)

Write a function *in C* called “myFunc” that will calculate and display the wind speed by reading the time of a rotation, scaling and then displaying the result using your hardware designed above.

- Assume that you have a function called **scale** that will convert the time of rotation to KPH. Scale accepts time per rotation in microseconds.
- Assume that you have another function called **display** that accepts KPH and displays the result. It is acceptable if the first speed value is incorrect. All subsequent values should be correct.
- Assume the APB bus runs at 1 MHz.

Function Prototypes:

```
void display(int kph);

int scale(int us);
```

Providing comments may improve your chance for partial credit.

```
void myFunc(){
    volatile uint32_t *timer_address = (volatile uint32_t*) 0x40050000;
    int time = *timer_address;
    int KPH = scale (time);
    display(KPH); //display the KPH
    *timer_address= 0; //reset the timer
    (uint32_t*)(0x4005004) = 1;

}
```

Part 3: “myFunc” (18 points)

Write a function *in C* called “myFunc” that will calculate and display the wind speed by reading the time of a rotation, scaling and then displaying the result using your hardware designed above.

- Assume that you have a function called **scale** that will convert the time of rotation to KPH. Scale accepts time per rotation in microseconds.
- Assume that you have another function called **display** that accepts KPH and displays the result. It is acceptable if the first speed value is incorrect. All subsequent values should be correct.
- Assume the APB bus runs at 1 MHz.

Function Prototypes:

```
void display(int kph);

int scale(int us);
```

Providing comments may improve your chance for partial credit.